Redes y Sistemas Complejos (2016-2017)
Grado en Ingeniería Informática
Universidad de Granada

Memoria Práctica 3
Estudio Comparativo de Métodos para Poda y Visualización de Redes.

Braulio Vargas López
DNI: 20079894C
Correo: brauliovarlop@correo.ugr.es

29 de noviembre de 2016

Índice

1. Poda y Visualización

En esta primera parte del guión, vamos a realizar un análisis de diez de las veinte redes disponibles, podándo cada una de las redes con la versión *BinaryPathfinder* del algoritmo. Las diez redes escogidas son:

1. France-2002
2. Germany-2002
3. Japan-2002
4. Spain-1996
5. Spain-1998
6. Spain-2002
7. Spain-2004
8. United_Kingdom-2002
9. United_States-2002
10. World

La razón de coger estas redes es por el hecho de poder ver la evolución de la producción científica en España a lo largo de ocho años, y poder compararlas con otros países durante el año 2002. Además, de poder ver la producción científica a nivel mundial en la última red.
A continuación, podemos ver los resultados obtenidos aplicando el algoritmo a cada uno de las redes seleccionadas:

France-2002

n=267	L	D
Red original	23986	0.675453
2	312	0.00878601
3	275	0.00774408
4	271	0.00763144
5	270	0.00760328
266	268	0.00754696

Germany-2002

n=269	L	D
Red original	25395	0.704516
2	313	0.00868335
3	277	0.00768463
4	272	0.00754591
5	270	0.00749043
268	269	0.00746269

Tabla 1.1: Tablas para las redes de France-2002.net y Germany-2002.net

Japan-2002

n=265	L	D
Red original	21754	0.621898
2	316	0.00903373
3	279	0.00797599
4	269	0.00769011
5	267	0.00763293
264	267	0.00763293

Spain-1996

n=243	L	D
Red original	5967	0.202938
2	394	0.0134
3	313	0.0106452
4	303	0.0103051
5	300	0.010203
242	300	0.010203

Tabla 1.2: Tablas para las redes Japan-2002.net y Spain-1996.net

Spain-1998		
n=258	L	D
Red original	12971	0.391247
2	320	0.00965222
3	279	0.00841553
4	267	0.00805357
5	267	0.00805357
257	267	0.00805357

Spain-2002		
n=264	L	D
Red original	21807	0.628154
2	320	0.00921765
3	274	0.00789261
4	265	0.00763337
5	263	0.00757576
263	263	0.00757576

Tabla 1.3: Tablas para las redes *Spain-1998.net* y *Spain-2002.net*.

Spain-2004		
(n=269)	L	D
Red original	24991	0.693309
2	332	0.00921045
3	280	0.00776785
4	272	0.00754591
5	271	0.00751817
268	270	0.00749043

United_Kingdom-2002		
(n=276)	L	D
Red original	28707	0.756443
2	326	0.00859025
3	288	0.00758893
4	280	0.00737813
5	279	0.00735178
275	276	0.00727273

Tabla 1.4: Tablas para las redes *Spain-2004.net* y *United_Kingdom-2002.net*.

United_States-2002		
(n=276)	L	D
Red original	31292	0.824559
2	314	0.00827404
3	287	0.00756258
4	279	0.00735178
5	277	0.00729908
275	275	0.00724638

World.net		
(n=218)	L	D
Red original	20154	0.85207
2	280	0.0118378
3	233	0.00985076
4	223	0.00942798
5	220	0.00930115
217	217	0.00917431

Tabla 1.5: Tabla para las redes *United_States-2002.net* y *World.net*.

Como podemos ver en las tablas anteriores, los resultados para todas las redes son parecidos. Con $n = 2$, el algoritmo poda una gran cantidad de enlaces en todas las redes, lo que hace bajar muchísimo la densidad de la red. Por ejemplo, en la Tabla **??**, para la red *France-2002.net* podemos ver cómo la red original tenía 23986 enlaces y una densidad $D \approx 0{,}7$, pasa a tener con 312 enlaces y una densidad de $D \approx 0{,}009$, unas 77 veces menor aproximadamente.
Además de esto, en todas las redes se da el mismo suceso y es que el número de enlaces para $n = 5$ y $n = \infty$ es prácticamente igual, o hay casos en los que es igual, como se puede ver en las tablas Tabla **??**. Esto se debe a que el algoritmo, a partir de este punto, no puede podar más.

1.1. Visualizaciones de las Redes

Las redes que vamos a visualizar son las redes de *Spain-1996.net* y *Spain-2004.net*. A continuación, podremos ver las distintas visualizaciones para los algoritmos ***Frutcherman & Reignold*** y ***Kamada & Kawai***. Además de esto, servirá para ver la evolución científica en España tras 8 años.
En ambas redes, podremos ver de color más rojo y más grande los nodos que más grado tienen. De igual manera pasa con los enlaces de la red. Serán de mayor grosor y más rojos cuanto mayor sea el peso del enlace.

1.2. Visualización para la red *Spain-1996.net*

(a) Distribución de nodos con Kamada & Kawai.

(b) Distribución de nodos con Frutcherman & Reignold.

Figura 1.1: Kamada & Kawai y Frutcherman & Reignold para la red Spain-1996 con $q = 2$.

En la Figura **??** podemos ver como el algoritmo Kamada & Kawai hace una distribución "similar" a un plano de metro, donde podemos ver de forma más clara, la distribución de la producción científica del país. En los nodos centrales (si el tamaño permitiera ver bien las etiquetas), podemos ver que los nodos más importantes son la medicina y la bioquímica, lléndose hacia la izquierda ramas científicas como la ingeniería, matemática aplicada... Sin embargo, Frutecherman & Reignold (Figura **??**) realiza una distribución circular, donde es más difícil interpretar los datos, ya que todos quedan más o menos a la misma distancia, cosa que con Kamada & Kawai no pasa, ya que aleja los nodos que menos relación tienen entre sí.

(a) Distribución de nodos con Kamada & Kawai.

(b) Distribución de nodos con Frutcherman & Reignold.

Figura 1.2: Kamada & Kawai y Frutcherman & Reignold para la red Spain-1996 con $q = 3$.

(a) Distribución de nodos con Kamada & Kawai.

(b) Distribución de nodos con Frutcherman & Reignold.

Figura 1.3: Kamada & Kawai y Frutcherman & Reignold para la red Spain-1996 con $q = 4$.

(a) Distribución de nodos con Kamada & Kawai.

(b) Distribución de nodos con Frutcherman & Reignold.

Figura 1.4: Kamada & Kawai y Frutcherman & Reignold para la red Spain-1996 con $q = n - 1$.

Como se puede ver en cada una de las imágenes podemos ver cómo, para este tipo de red, el algoritmo Kamada & Kawai realiza una mejor visualización que Frutcherman & Reignold, dejando más alejadas las materias que no tienen relación entre sí, y más cercanas las que sí tienen relación, mientras que Frutcherman & Reignold, realiza una visualización circular, al querer dejar los enlaces con la misma longitud.

1.3. Visualización para la red *Spain-2004.net*

(a) Distribución de nodos con Kamada & Kawai.

(b) Distribución de nodos con Frutcherman & Reignold.

Figura 1.5: Kamada & Kawai y Frutcherman & Reignold para la red Spain-2004 con $q = 2$.

En esta red, pasados 8 años desde la anterior, podremos observar la evolución de la producción científica, tomando mucha más fuerza la medicina, y materias como la Neurociencia, la Psicología, Genética, y otros campos relacionados con las Física y la Ingeniería Electrónica, entre otras. Además de esto, en la red original, se ve incrementado casi el doble el número de nodos existentes y un gran aumento en el número de enlaces de la red, observándose cómo aumenta la producción científica de España a lo largo de esos 8 años, a parte de ver cómo van cambiando los principales temas de investigación.

(a) Distribución de nodos con Kamada & Kawai.

(b) Distribución de nodos con Frutcherman & Reignold.

Figura 1.6: Kamada & Kawai y Frutcherman & Reignold para la red Spain-1996 con $q = 3$.

(a) Distribución de nodos con Kamada & Kawai.

(b) Distribución de nodos con Frutcherman & Reignold.

Figura 1.7: Kamada & Kawai y Frutcherman & Reignold para la red Spain-1996 con $q = 4$.

(a) Distribución de nodos con Kamada & Kawai.

(b) Distribución de nodos con Frutcherman & Reignold.

Figura 1.8: Kamada & Kawai y Frutcherman & Reignold para la red Spain-1996 con $q = n - 1$.

Al igual que pasaba en la sección anterior, podemos ver que el algoritmo Kamada & Kawai realiza para este caso una visualización muchísimo mejor que el algoritmo Frutcherman & Reignold, ya que deja más claro cuáles son los nodos más importantes de la red, realizando esa visualización similar a una línea de metro, mientras que el otro realiza una visualización circular, que para este caso, no deja muy clara la información que transmite la red.

2. Análisis de eficiencia de las variantes del algoritmo *Pathfinder*

Para analizar la eficiencia práctica de los distintos algoritmos, vámos a lanzarlos para redes aleatorias de tamaño 500, 1000, 2000, 5000 y 10000 nodos. Para ello, se ha creado un script

que ejecuta los algoritmos para las distintas redes y genera una tabla en la que podemos ver los resultados de las distintas ejecuciones.

N	$\|L\|$	$\|D\|$	Original	Binary	Fast	MST (Teórico)	MST (Práctico)
500	1252	0.01003	37,71665	4,40057	0,29051	0,00333	0,00296
1000	4999	0.01000	617,89157	49,98381	2,22931	0,01109	0,01079
2000	20041	0.01002	> 1800	750,72829	18,83852	0,17109	0,10172
5000	124525	0.00996	> 1800	> 1800	276,64002	0,53278	0,48843
10000	498666	0.00997	> 1800	> 1800	> 1800	2,53242	2,49397

Tabla 2.1: Tabla de resultados para los distintos algoritmos.

Como se puede ver en la Tabla **??**, la versión original del algoritmo Pathfinder, es bastante ineficiente, en comparación con el resto, ya que para problemas de pequeño tamaño, ya requiere una gran cantidad de tiempo y un gran espacio. Esto se debe, al estar basado en programación dinámica, tienen unos ordenes de complejidad en tiempo de $\mathcal{O}(n^4)$ y una complejidad en espacio de $\mathcal{O}(n^3 - 2 \cdot n^2) \equiv \mathcal{O}(n^3)$, que aunque sea polinómica, para problemas donde n, ni siquiera es grande, con un $n = 1000$ los tiempos de ejecución son del orden de unos 10 minutos en una máquina con un procesador que funciona con *overclock* a $4{,}6GHz$.
La segunda versión del algoritmo Pathfinder, *BinaryPathfinder*, mejora estos órdenes de complejidad del algoritmo, pero aun así, para problemas de gran tamaño, sigue sufriendo y es incapaz de acabar en tiempos razonables. Esto se debe a que, a pesar de haber rebajado los órdenes de complejidad, estos siguen siendo $\mathcal{O}(n^3 \log n)$ para el tiempo, y $\mathcal{O}(4 \cdot n^2)$ para el espacio, que aunque sean menores, siguen siendo "malos" para problemas donde n es grande.
En la versión *FastPathfinder*, se vuelve a reducir tanto la complejidad en tiempo como en espacio, pasando a tener $\mathcal{O}(n^3)$ para el tiempo, y $\mathcal{O}(3 \cdot n^2)$ para el espacio. Esta reducción permite que para problemas donde n es pequeña, pase a tiempos de décimas de segundo, y nos permite ejecutar problemas donde $n = 5000$, en unos 5 minutos aproximadamente. Aun así, a pesar de esta mejora, seguimos poder sin ejecutar problemas donde la n tiene un tamaño medio en un tiempo razonable, es decir, $n > 7000$.
Estas tres versiones utilizan la programación dinámica como algoritmo base, lo que provoca estos órdenes de complejidad tanto en tiempo como en espacio. Las siguientes versiones cambian de paradigma, abandonando la programación dinámica y usando un algoritmo de la teoría clásica de grafos: **el algoritmo de Kruskal**.
El algoritmo de Kruskal es un algoritmo clásico de la teoría de grafos, muy utilizado para encontrar el árbol generador minimal de un grafo G que toma como entrada, ya que la complejidad en tiempo y espacio es muy pequeña, lo que lo hace muy eficiente. Tomando esto como base, surgieron los siguientes algoritmos.
La versión teórica de esta versión, *MST-Theorical*, es exactamente igual a la versión *MST-Practical* pero cambia la estructura de datos utilizada en los algoritmos. La versión teórica tiene un orden de complejidad menor que la versión práctica, pero los tiempos de ejecución son mayores debido a la estructura de datos que tiene asociada. La versión teórica usa una estructura de datos de conjuntos disjuntos con una unión por rango y comprensión de caminos, mientras que la versión práctica, utiliza una lista indexada de conjuntos disjuntos, es decir, tres listas de enlaces con sus pesos (F, T y H) junto con el índice. Para más detalle, podemos consultar el artículo original [**?**].
Los órdenes de complejidad son $\mathcal{O}(n^2 \cdot \log n)$ para el tiempo y $\mathcal{O}(3 \cdot n^2 + n)$ para el espacio en la versión teórica del algoritmo MST, mientras que para la versión teórica tenemos $\mathcal{O}(n^3)$ para el tiempo y $\mathcal{O}(3 \cdot n^2 + n)$ para el espacio. Como podemos ver, la versión teórica tiene unos órdenes de complejidad menor que la versión práctica, pero aun así, gracias a que las estructuras de datos son más simples en la versión práctica, podemos observar en la Tabla **??** la versión práctica es más rápida que la teórica.
A pesar de la mejora en tiempo, supone un deterioro en los órdenes de complejidad en espacio, pero es un precio a pagar para poder representar redes con un mayor número de nodos en un tiempo razonable.

2.1. Visualización de las redes aleatorias

A continuación podemos ver algunas visualizaciones para las redes de tamaño 2000, 5000 y 3000 con las siguientes visualizaciones:

- **Radial Layout**.
- **Force Atlas 2**.
- **OpenORD**.

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución de nodos generada con RadialLayout.

Figura 2.1: Red de tamaño $n = 2000$

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución de nodos generada con RadialLayout.

Figura 2.2: Red de tamaño $n = 5000$

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución de nodos generada con RadialLayout.

Figura 2.3: Red de tamaño $n = 10000$

En ellas podemos ver las distintas visualizaciones para los distintos algoritmos, siendo el más vistoso el algoritmo *RadialLayout*, aunque, la capacidad para transimitir información de este último para estas redes es más bien poca. En las otras visualizaciones, al menos se guarda algo de información de los nodos más importantes de la red, y su distribución en esta, pero no mucho más, ya que se ha buscado más la estética que el representar información.

3. Extras opcionales

3.1. Otras visualizaciones

A continuación, podemos ver otras visualizaciones distintas a las que se pueden obtener con los algoritmos de Kamada & Kawai y Frutecherman & Reignold, con las mismas redes que teníamos en la primera parte de la práctica. Uno de estos algoritmos es **OpenORD**, que está basado en el algoritmo de Frutecherman & Reignold, pero con la intención de diferenciar más los clusters. Este algoritmo ejecuta durante un número fijo de iteraciones un Enfriamiento Simulado para agrupar los clusters y diferenciarlos en la pantalla, eliminando las aristas más largas para permitir diferenciar los clusters. También podremos ver la visualización que genera el algoritmo **ForceAtlas 2**, una variante del algoritmo Kamada & Kawai, capaz de manejar redes más grandes y realizar visualizaciones de calidad. La diferencia con Kamada & Kawai es que cambia los parámetros de *atracción* y *repulsión* por un parámetro de *escalado*. Además de esto, podremos ver el algoritmo de **YifanHu**, que realiza unas visualizaciones de muy buena calidad para las redes que estamos analizando, donde quedan muy bien definidas las distintas áreas principales y las áreas que tienen relación con estas, aunque, su capacidad de dibujado es menor que en los anteriores.

3.1.1. Visualización para la red *Spain-1996.net*

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución generada con el algoritmo YifanHu.

Figura 3.1: ForceAtlas, OpenORD y YifanHu para la red Spain-1996 con $q = 2$.

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución generada con el algoritmo YifanHu.

Figura 3.2: ForceAtlas, OpenORD y YifanHu para la red Spain-1996 con $q = 3$.

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución generada con el algoritmo YifanHu.

Figura 3.3: ForceAtlas, OpenORD y YifanHu para la red Spain-1996 con $q = 4$.

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución generada con el algoritmo YifanHu.

Figura 3.4: ForceAtlas, OpenORD y YifanHu para la red Spain-1996 con $q = n - 1$.

3.1.2. Visualización para la red *Spain-2004.net*

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución generada con el algoritmo YifanHu.

Figura 3.5: ForceAtlas, OpenORD y YifanHu para la red Spain-2004 con $q = 2$.

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución generada con el algoritmo YifanHu.

Figura 3.6: ForceAtlas, OpenORD y YifanHu para la red Spain-2004 con $q = 3$.

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución generada con el algoritmo YifanHu.

Figura 3.7: ForceAtlas, OpenORD y YifanHu para la red Spain-2004 con $q = 4$.

(a) Distribución de nodos con ForceAtlas 2.

(b) Distribución de nodos con OpenORD.

(c) Distribución generada con el algoritmo YifanHu.

Figura 3.8: ForceAtlas, OpenORD y YifanHu para la red Spain-2004 con $q = n - 1$.

Como podemos ver, los tres algoritmos ofrecen buenas visualizaciones para este tipo de redes, donde es importante ver que las áreas que estén relacionadas estén juntas, mientras que las que menos relacionadas están se sitúen más lejos en la visualización. Además, son algoritmos que funcionan bastante bien para el tamaño de red que estamos utilizando.

3.2. Aumentando el tamaño de la red

A continuación, en la Tabla **??** podemos ver la tabla de resultados que vimos anteriormente (Tabla **??**, pero con el añadido de que el tamaño de n se ha extendido hasta los 30000 nodos.

N	$\|L\|$	$\|D\|$	Original	Binary	Fast	MST (Teórico)	MST (Práctico)
500	1252	0.01003	37,71665	4,40057	0,29051	0,00333	0,00296
1000	4999	0.01000	617,89157	49,98381	2,22931	0,01109	0,01079
2000	20041	0.01002	> 1800	750,72829	18,83852	0,17109	0,10172
5000	124525	0.00996	> 1800	> 1800	276,64002	0,53278	0,48843
10000	498666	0.00997	> 1800	> 1800	> 1800	2,53242	2,49397
20000	1.99927e+06	0.00999686	> 1800	> 1800	> 1800	15,35867	15,48830
30000	4.49748e+06	0.00999473	> 1800	> 1800	> 1800	47,23194	47,23149

Tabla 3.1: Tabla de resultados para los distintos algoritmos.

En esta tabla, podemos comprobar como para tamaños de n grandes, los algoritmos MST ya empiezan a tomar un tiempo algo considerable para finalizar la ejecución, pero sigue siendo los únicos capaces de ejecutarlo en un tiempo razonable, mientras que los otros, son totalmente incapaces de hacerlo, debido a la propia base del algoritmo (programación dinámica) y a las estructuras de datos asociadas que utilizan para resolver el problema.
También en la tabla, podemos observar cómo el tiempo del MST-Teórico es menor que el MST-Práctico. Esto se debe principalmente a que pueden haberse visto afectados los tiempos por interrupciones del sistema y haber salido un poco más rápido para tamaño $n = 20000$. Para tamaño $n = 30000$, se producen tiempos iguales, lo que concuerda más con el análisis visto anteriormente. Esto se produce a que para un tamaño de $n = 30000$, con $16GB$ de memoria RAM, el sistema operativo hace operaciones de swapping en disco, por lo que afecta a los tiempos de ejecución.